Developing An Effective Metrics Program

L. Rosenberg, Ph.D.,

Unisys /GSFC, Bld 6 Code 300.1,

Grenebelt, MD 20771 USA,

Tel:301-286-0087,

Fax: 301-286-0304

L. Hyatt,

GSFC NASA Bld 6 Code 302.

Greenbelt, MD 20771 USA

Tel:301-286-7475,

Fax: 301-286-1701

ABSTRACT

Software metrics programs can be very cost effective or a total waste of resources. This paper discusses how to develop an effective, affordable metrics program that will help project managers monitor project risks and evaluate product quality. The Goal/Question/Metric paradigm (GQM) is used to demonstrate how a meaningful metrics program can be started and uses data from projects at Goddard Space Flight Center (GSFC) to demonstrate some analysis and application techniques.

This paper supplies both project managers and software developers with techniques to initiate a metrics program that yields timely, relevant, usable information at minimal cost.

1. INTRODUCTION

The Software Assurance Technology Center (SATC) was established in 1992 in the Systems Reliability and Safety Office at NASA's Goddard Space Flight Center (GSFC). The SATC has programs in four areas: Software Standards and Guidebooks; Software Metrics Research and Development; Assurance Tools and Techniques; and Project Support and Outreach. The SATC, as a center of excellence in software assurance, is dedicated to making measurable improvement in the quality and reliability of software developed for GSFC and NASA.

But developing a metrics program is not easy. It has many possible pit-falls that can lead to the ruin of the metrics program itself and possibly the project if incorrectly applied. This paper starts with a discussion of the costs versus benefits of a metrics program. How to develop and implement a metrics program using the Goal/Question/Metric Paradigm is then discussed, followed by an example using data from GSFC projects.

2. METRIC PROGRAM COSTS VS. BENEFITS

It is difficult to pin down the costs of a metrics program because metrics are usually just one aspect of an overall improvement program. When investigating the feasibility of starting a metrics program, it is often found that managers are individually collecting some form of data. This decreases initial program start up cost. Accurate and complete measurements are not inexpensive; comprehensive metrics programs for software products and process annual costs can be 2 to 3 percent of the total software budget for collecting hard data.[31 Attempts to pin down the cost of metrics hide the real issue, however, developers don't really have a choice. The cost of not implementing a software metrics program can be measured in terms of project and business failures. Those projects and companies who make the investment in metrics have a competitive advantage over those who do not. They have the advantage of more informed and timely decisions that will ultimately make them more successful, with the best track records in terms of bringing software projects to completion and achieving high levels of user satisfaction.[2]

It is difficult, if not impossible, to place a dollar amount on the benefits of a metrics program because as in the case of risk management, you are trying to measure something that did not happen. The benefits derived are also not only applicable to the current project but to future projects. As with any new project, whether it is implementing a new engineering design or a metrics program, start up costs are high. But as management and staff become familiar with the tasks and tools are developed, the costs decrease to a low maintenance level.

3. DEVELOPING A METRICS PROGRAM

3.1 Where to Start

Once a developer decides to implement a metrics program, the next step is How. How a metrics program is developed can determine its success or failure. One approach is to investigate tools available for metrics collection, purchase the tool, then collect and attempt to apply whatever metrics are provided by the tool. This may work but has a major hurdle - what will the data collected tell the management and developers about their specific project. Data collected just because it is available has minimal value at best and usually ends up a waste of resources.

Successful metrics programs generally begin by focusing on a problem. At the start of the metrics program, goals must be established that address the problem. Related questions that management wants answered are identified then the data that is needed to answer these questions are specified. This leads to the tool specification for purchase or in-house development. Data collection can be expensive if not carefully monitored - the temptation is to collect all possible data and decide how to use it later.

The earlier benefits are seen by both management and developers, the sooner metrics programs are accepted. Metric programs should be designed to show visible benefits as soon as possible, this is the key to continued support.

3.2 Goal/Question/Metric (GQM) Paradigm

The Goal/Question/Metric (GQM) Paradigm is a mechanism that provides a framework for developing a metrics program. It was developed at the University of Maryland as a mechanism for formalizing the tasks of characterization, planning, construction, analysis, learning and feedback. The GQM paradigm was developed for all types of studies, particularly studies concerned with improvement issues. The paradigm does not provide specific goals but rather a framework for stating goals and refining them into questions to provide a specification for the data needed to help achieve the goals.[l]

The GQM paradigm consists of three steps:

1. 1. Generate a set of goals
   2. Derive a set of questions
   3. Develop a set of metrics

1 - Generate a set of goals based upon the needs of the organization- Determine what it is you want to improve. This provides a framework for determining whether or not you have accomplished what you set out to do. Goals are defined in terms of purpose, perspective and environment using generic templates:

Purpose: To (characterize, evaluate, predict, motivate, etc.)the (process, product, model, metric, etc.) in order to (understand, assess, manage, engineer, learn, improve, etc.) it.

Perspective: Examine the (cost, effectiveness, correctness, defects, changes, product metrics, reliability, etc.) from the point of view of the (developer, manager, customer, corporate perspective, etc.)

Environment: The environment consists of the following: process factors, people factors, problem factors. methods, tools, constraints, etc.

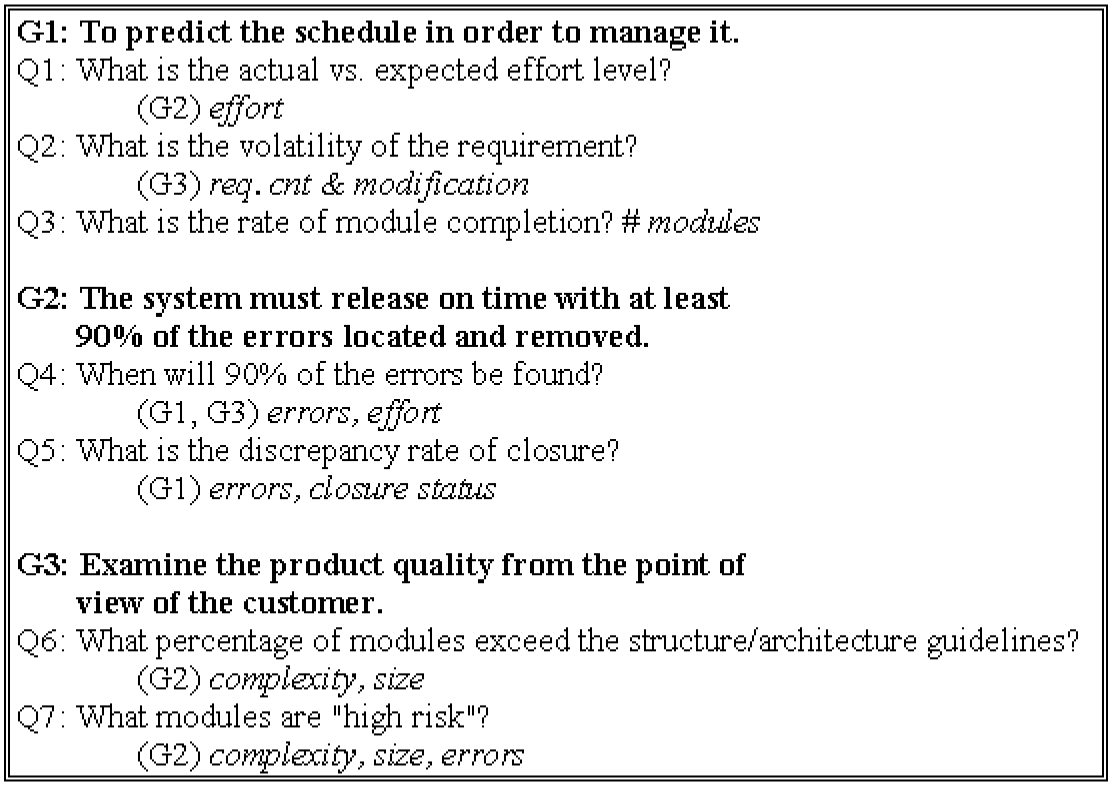
2 - Derive a set of questions - The purpose of the questions is to quantify the goals as completely as possible. This requires the interpretation of fuzzy terms within the context of the development environment. Questions are classified as product-related or processrelated and provide feedback from the quality perspective. Product-related questions define the product and the evaluation of the product with respect to a particular quality (e.g., reliability, user satisfaction). Process-related questions include the quality of use, domain of use, effort of use, effect of use and feedback from use.

3 - Develop a set of metrics and distributions that provide the information needed to answer the questions - In this step, the actual data needed to answer the questions are identified and associated with each of the questions. As data items are identified, it must be understood how valid the data item will be with respect to accuracy and how well it responds to the specific question. The metrics should be objective and subjective and should have interpretation guidelines, i.e., what value of the metric specifies the product higher quality. Generally, a single metric will not answer a question, but a combination of metrics is needed.
Once goals are defined, questions derived, and metrics developed, matrices are created to indicate their relationships and to identify single relationships that may not be cost effective.

3.3 GQM Example

The most effective way to understand a methodology is to review an example. This section demonstrates how a small metrics program would be developed using the GQM. The program starts with the goals, questions and proposed metrics, then demonstrates how GSFC data could be used to answer some of the questions and satisfy the goals.

Figure 1 demonstrates sample goals, questions and metrics. The goals are general and could be adapted with minor modifications to any project development. Questions are derived to quantify the goals, often supporting more than one goal as shown in parenthesis( ). The metrics needed to provide the answers to the questions are then chosen and shown in italics.



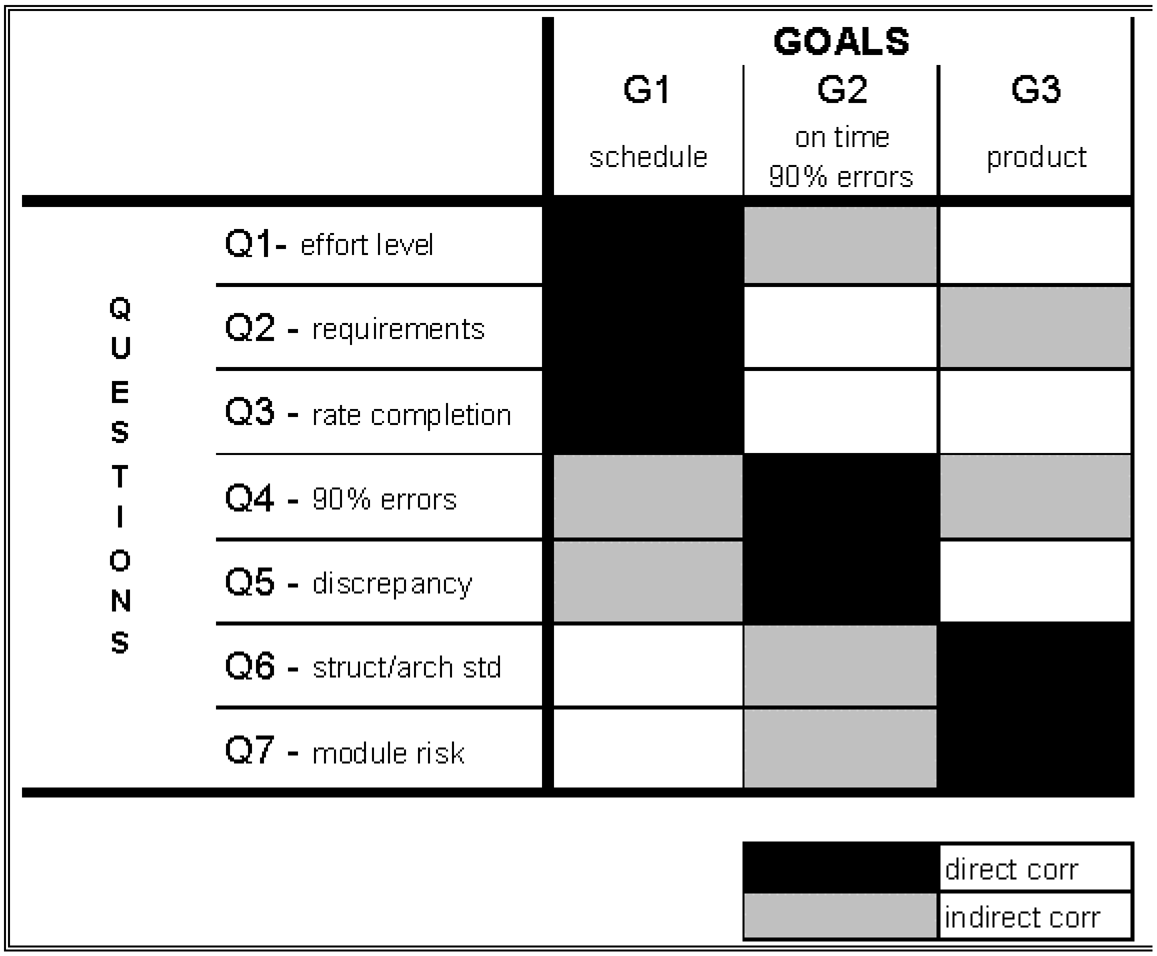
G1: To predict the schedule in order to manage it.
Q1: What is the actual vs. expected effort level?
(G2) *effort*
Q2: What is the volatility of the requirement?
(G3) *req. cnt & modification*
Q3: What is the rate of module completion? # *modules*

G2: The system must release on time with at least 90% of the errors located and removed.
Q4: When will 90% of the errors be found?
(G1, G3) *errors, effort*
Q5: What is the discrepancy rate of closure?
(G1) *errors, closure status*

G3: Examine the product quality from the point of view of the customer.
Q6: What percentage of modules exceed the structure/architecture guidelines?
(G2) *complexity, size*
Q7: What modules are "high risk"?
(G2) *complexity, size, errors*

Figure 1: Goals/Questions/Metrics

Matrices similar to Figure 2 are developed showing direct and indirect correlations between goals and questions, then questions and metrics.



QUESTIONS	GOALS G1 schedule	G2 on time 90% errors	G3 product
Q1- effort level	■	▒	
Q2 - requirements	■		▒
Q3 - rate completion	■		
Q4 - 90% errors	▒	■	▒
Q5 - discrepancy	▒	■	
Q6 - struct/arch std		▒	■
Q7 - module risk		▒	■

■	direct corr
▒	indirect corr

Figure 2: Goals -> Questions

In the remainder of the paper, we will demonstrate some of the metrics and data that can be used to answer the questions and satisfy the goals.

Question 1: Expected vs. actual effort level

Effort is usually measured in hours worked on specific project tasks, such as training, requirements, design, coding, and testing. In Figure 3, the Rayleigh Manpower Curve for effort expenditure for a typical software project is plotted against some project effort data. [4] Projects ramp up to full speed fairly quickly, then taper off as the maintenance phase approaches. Applying this curve assists managers adjust personnel levels to the expected work load.

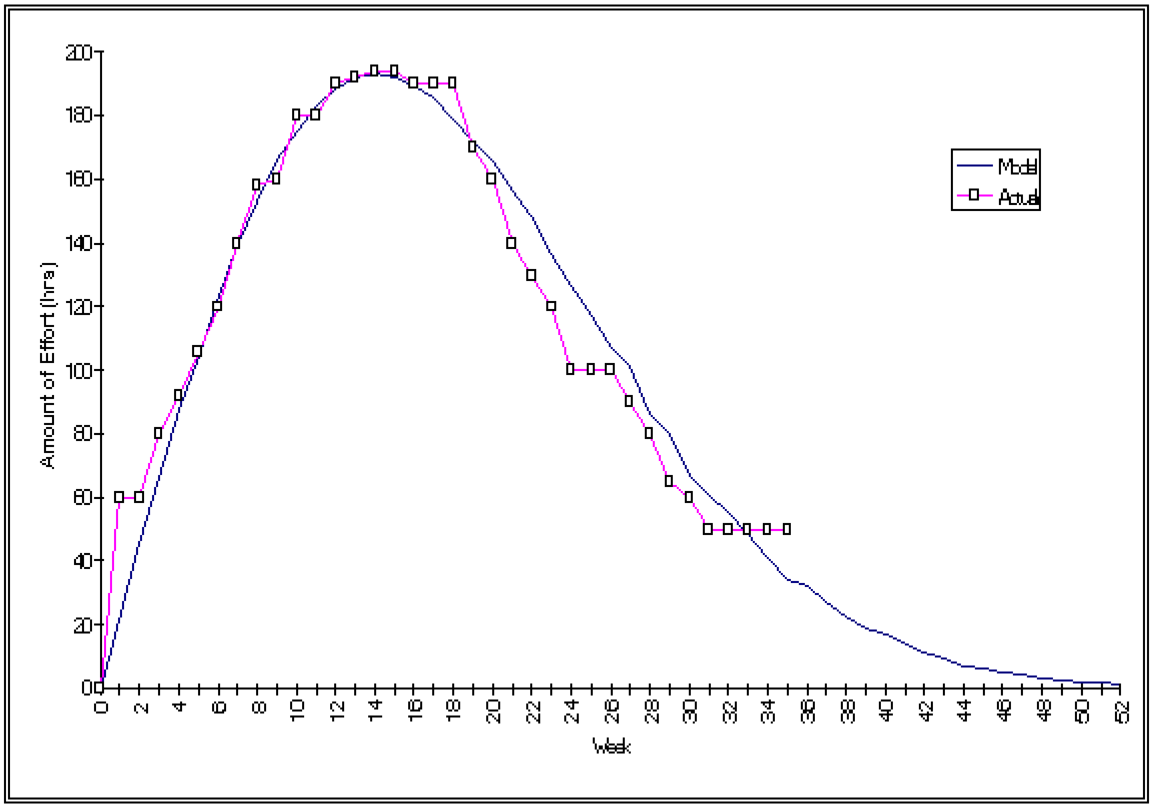


Figure 3: Rayleigh Manpower Curve

Question 2: Requirement volatility

Late requirement changes are costly and may cause a ripple effect and additional changes. The earlier in the Life cycle the requirements stabilize, the less the risk. Figure 4 shows total requirements per schedule, indicating stabilizing requirements. Modifications should also be tracked.

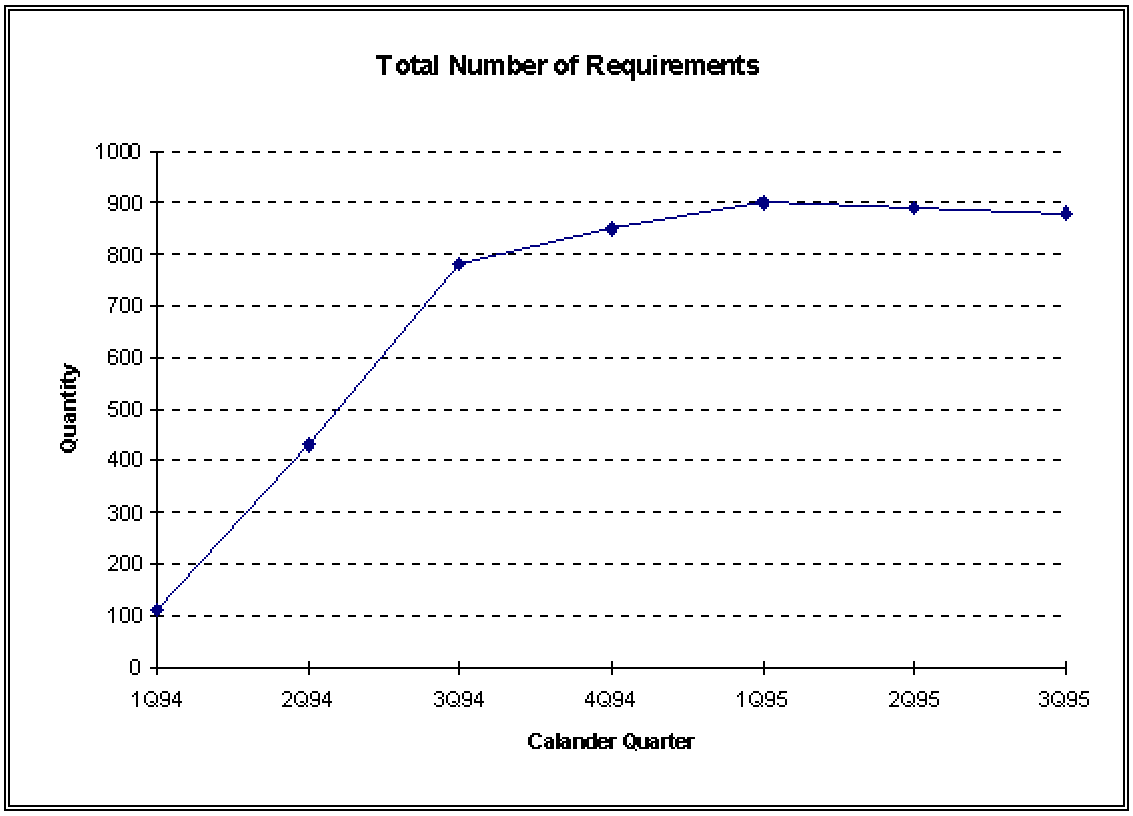


Figure 4: Stabilization of Requirements

Question 4: 90% of the errors be located.

This implies the ability to estimate the total number of errors in the software. One industry guideline is to expect approximately7 errors per 1000 Source Lines of Code. This guideline is helpful in an overall estimate of the number of errors, but does not take into account the rate at which errors are removed. The SATC is working to release the Waterman Error Trending Model for determining the status of testing by projecting the number of errors remaining in the software and the expected time to find some percentage of errors. The SATC is developing this model rather than using the standard Musa model because it provides for nonconstant testing resource levels and is less sensitive to data inaccuracy. Figure5 is an example of the model's application.

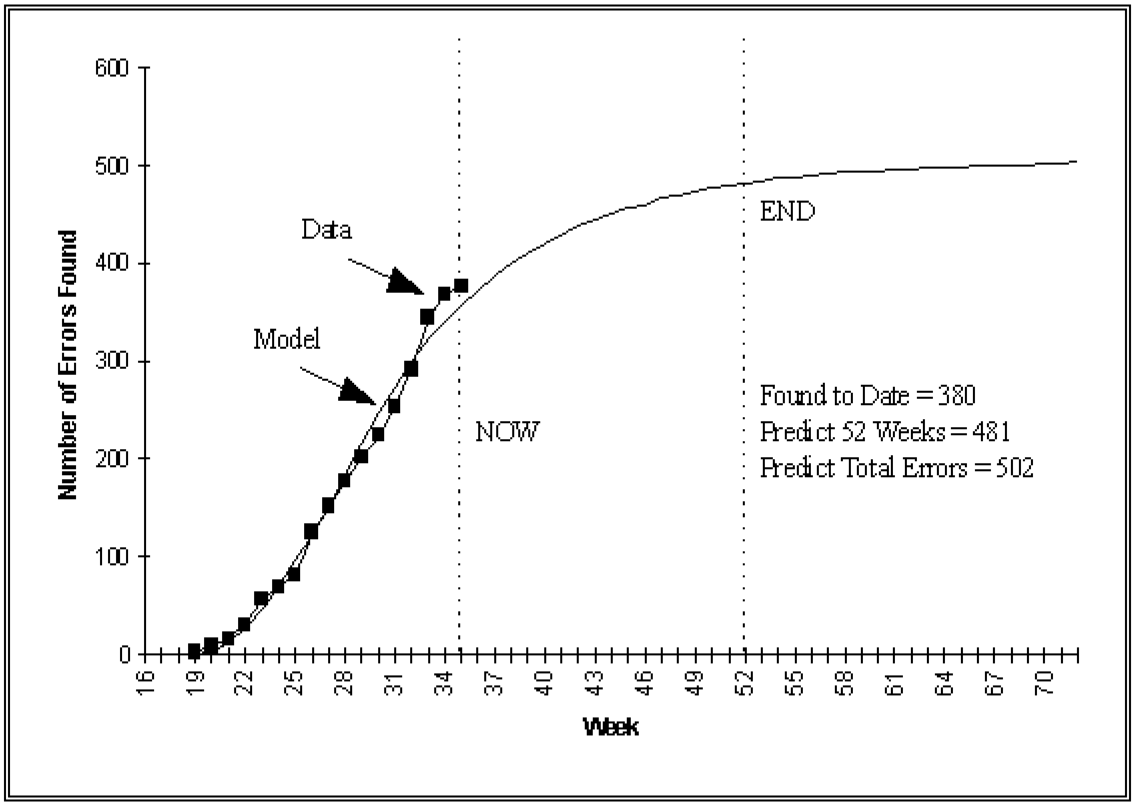


Figure 5: Waterman Error Trending Model

If this project is expected to release at week 52, Figure 5 indicates that 96% of the errors will be found.

Question 6: Modules exceeding guidelines

Figure 6 is a graph template developed by the SATC to use as an indicator of the module risk levels. The x-axis represents the number of executable statements in a module; the y-axis is the extended cyclomatic complexity (number of test paths) for the module.

There are many different guidelines for both code measures as to when risk increases or decreases and what is are acceptable levels. The parameters in Figure 6 are based on guidelines from various industry, military and NASA sources as well as error correlations from GSFC data.

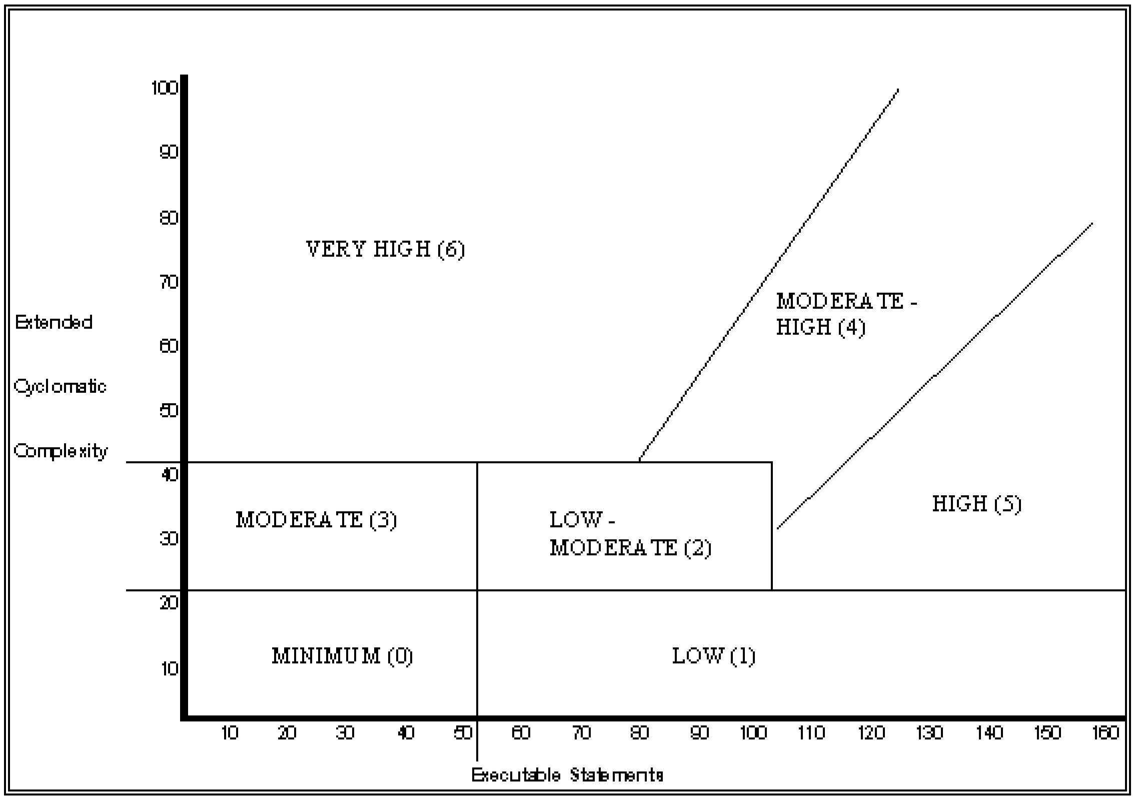


Figure 6: Classification of Module Risk

To apply the graph in Figure 6, each module of code is plotted as shown in Figure 7. The percentage of modules in each region and a list of module names in Regions 4, 5, and 6 are supplied to the project management. It is recommended that developers further investigate the modules in these regions using further using metrics such as fan in/ fan out, comment percentage and number of errors. One observation made by the SATC is that C and C++ code have a lower percentage of modules in Regions 4, 5 and 6 than FORTRAN.

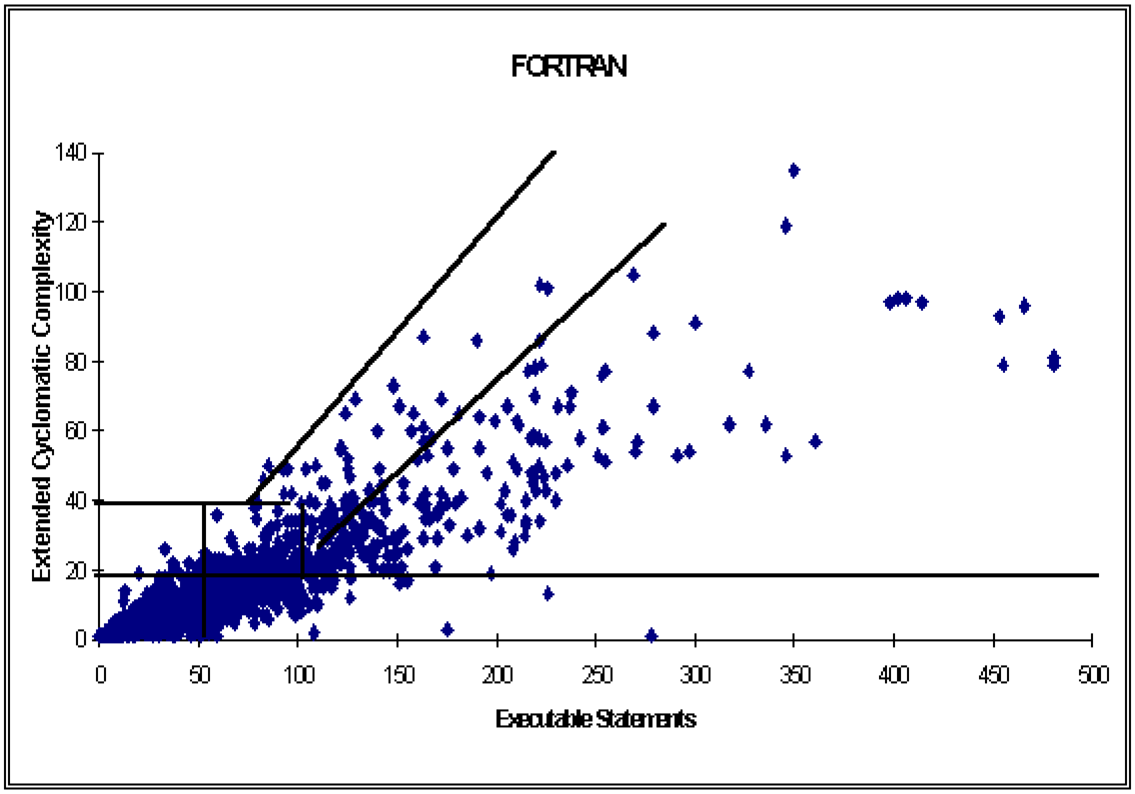
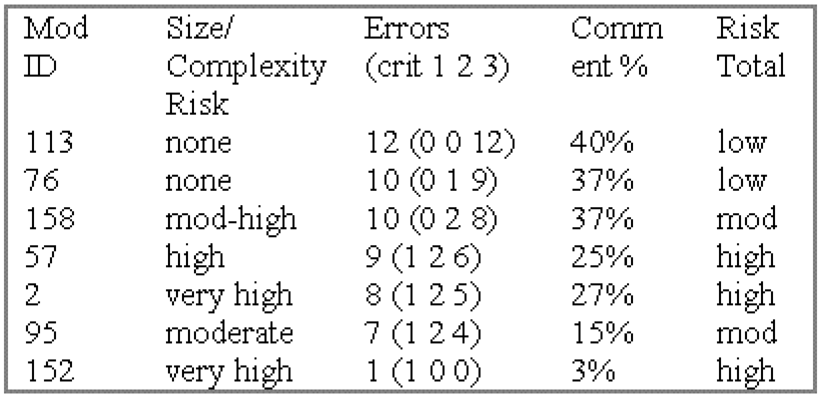


Figure 7: Modules at Risk in a FORTRAN Project

Question 7: High risk modules

To answer question 7, inputs from multiple metric are needed. One factor is the risk derived from Table 3. The number of errors by criticality also serves as an input. Comment percentage also supplies information relative to risk. Factors such as fan-in/fan-out and internal data flow also influences risk but are not shown in Table 3.



Mod ID	Size/ Complexity Risk	Errors (crit 1 2 3)	Comm ent %	Risk Total
113	none	12 (0 0 12)	40%	low
76	none	10 (0 1 9)	37%	low
158	mod-high	10 (0 2 8)	37%	mod
57	high	9 (1 2 6)	25%	high
2	very high	8 (1 2 5)	27%	high
95	moderate	7 (1 2 4)	15%	mod
152	very high	1 (1 0 0)	3%	high

Table 3: Risk evaluation of code modules

4. CONCLUSION

The SATC applies goals to evaluate the quality of products (requirement documents through test applications) and provide risk information to project managers. Metric programs are initiated to answer a question or provide numerical input to solve a problem.

The first step in developing a metrics program is to identify what are the goals or objectives of the program, then stay focused on them. The objectives can be expanded into specific goals using the structure of the Goal/Question/Metric templates.

The second step is to define the T attributes that are be measured. These attributes are a subset of the quality attributes and are chosen based on the project objectives and goals. If the GQM isused, some of the goals will relate to the attributes.

The third step in developing the metrics program is to clarify and quantify the goals. This is done by specifying questions and identifying metrics and data that is needed. At this point a tool is chosen based on the needs of the project.

The final and a very critical step is to close the loop - provide management with answers to their questions based on the metric analysis. The key to continued success of a metrics program is immediate, visible benefits. It must do the job it was designed to do and supply management with usable information to solve their current problem in a timely fashion.

REFERENCES

[1] Basili, Victor R, and Rombach, H. Dieter,
"Tailoring the Software Process to Project Goals and Environments", Department of Computer Science, University of Maryland, ACM, 1987.

[2] Grady, Robert, Practical Software Metrics For Project Management and Process Improvement, Prentice Hall, 1992.

[3] Gillies, Alan, Software Quality, Theory and Management, Chapman & Hall Computing, 1992.

[4] Putnam, L., Myers, W., Measures for Excellence: Reliable Software on Time, Within Budget, Yourdin Press, 1992.

Presented at the European Space Agency Software Assurance Symposium, Netherlands, March, 1996.